COST REDUCTION VIA COMMON ONBOARD SOFTWARE

John A. Rohr
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California USA
John.A.Rohr@Jpl.Nasa.Gov

The Jet Propulsion Laboratory has been building, launching, and operating deep-space probes for more than 30 years. Initial JPL probes were sent to the moon in the 1960's. Subsequent probes flew past nearby planets Mercury, Venus, and Mars. In the 1970's, two Viking probes landed on Mars and two Voyager spacecraft began their journey to the edge of the Solar System. In the 1980's and 1990's, other probes have been sent toward the Sun, planets, and outer space to increase our scientific knowledge of the Solar System.

Each space mission requires an onboard commanding and sequencing capability to direct the spacecraft through various actions to accomplish the scientific and engineering tasks required by the mission objectives. Early JPL spacecraft used programmable sequencers. These devices were hard-wired logic machines which could accept specific parameters. The first special-purpose computing capability consisted of a computer with a limited memory of 128 words and a set of 16 special-purpose, two-address instructions. The same computer was later enhanced by expanding the memory to 512 words. All onboard commanding and sequencing was accomplished within these constraints.

The first JPL spacecraft to have a general-purpose computer were the Viking Orbiter spacecraft. These spacecraft carried two identical computers which implemented the Computer Command Subsystem. Each computer had a 4,096-word memory and a set of 64 instructions, a few of which were dedicated to operating the custom input-output units which were included. Also onboard were dual telemetry computers and dual attitude control computers. Computers almost identical to the Viking Orbiter computers are used on the Voyager spacecraft.

The software used for commanding and sequencing onboard the Viking and Voyager spacecraft utilizes an onboard interpreter driven by a set of tables generated by ground software. The tables implement a language called Virtual Machine Language (VML). This language is used to program the spacecraft to accomplish the scientific and engineering tasks required by the mission objectives. VML is a very efficient language. All of the tables required for commanding and sequencing the spacecraft fit in about 2000 words in each computer memory. These words are reloaded as required from the ground.

The Galileo spacecraft utilizes a Command and Data System which is based on a distributed set of six RCA 1802 computers. This is a significantly different architecture from that of Viking and Voyager. The total amount of memory is approximately 500,000 words. The commanding and sequencing language for Galileo is very different from that of Viking and Voyager, but it is also called VML (for Virtual Machine Language). The Galileo VML implements all the capabilities required for commanding and sequencing onboard the Galileo spacecraft.

The Cassini spacecraft is using an IBM 1750A (GVSC) for its onboard computing capability. This is yet another architecture and yet another, different VML has been invented for the mission. Other missions have also used VML's which were different from those of previous missions.

The design and use of a different VML for each JPL mission has been expensive both in terms of the design and implementation effort for the language and in terms of the effort required to develop a ground system to support the language. Even though different hardware has been used, many of the functions required onboard are the same. In the future, multiple missions should be able to share one common language for onboard commanding and sequencing.

The use of a common language for onboard commanding and sequencing should reduce the cost of developing flight software for new missions. The amount of new flight code which would be needed should be minimized. Furthermore, use of the same language as other missions would minimize training needed for mission programmers. Having learned the language for one mission, only minimal retraining should be needed to use the language on another mission.

The use of a common language for onboard commanding and sequencing should also reduce the cost of developing ground software for new missions. JPL has invested a considerable effort over recent years to develop a multimission ground system. The use of a common language onboard for multiple missions would further help to reduce costs by eliminating the need to develop new software to support new onboard languages. Software developed to support a common onboard language would require only minimal modification, if any, from mission to mission. Also, all personnel would know the common onboard language and would be able to work on multiple missions or transition from one mission to another without needing to learn another onboard language.

The use of a commercial language as the common onboard language could provide several advantages for JPL. First and foremost, JPL would not need to develop and support a new language for each new mission. Second, the use of a commercial language would allow JPL to take advantage of capabilities in the language not available in the various VML's which have been used on prior JPL missions. Third, maintenance of the language would be the responsibility of the vendor, not JPL, thus saving JPL the cost of maintenance of the language. There are some potential problems, however, which must be considered if a commercial language is to be used for JPL missions. It is likely that some adaptation of both the JPL multimission software and the commercial language itself may be required for all components to work together effectively in the JPL operations environment. Also, JPL will need to ensure that the language has all the capabilities required to support JPL missions on which it would be used.

One commercial language which has been of interest to JPL mission personnel as a possible candidate for a common onboard command and control language is the Spacecraft Command Language (SCL), a product of Interface & Control Systems, Inc. of Columbia, MD. SCL was included onboard the Clementine spacecraft flown by the Naval Research Laboratory (NRL). Although the use of SCL on Clementine was limited, the capabilities of the language were demonstrated on an actual flight.

Because of the interest in SCL at JPL, a study was conducted in 1994 to determine the feasibility of using SCL as a common onboard language for command and control of JPL spacecraft. The study team consisted of personnel from several different areas at JPL including both flight and ground systems. As the study progressed, three groups were formed within the team to address overall requirements, flight software issues, and ground software issues. Near the end of the study, a group consensus indicated the need for direct input to the study from the implementers and users of SCL. Thus, two individuals who had worked with SCL on Clementine joined to study team for one week of intensive work. They contributions were very helpful to the team. At the completion of the study, a report was issued which gave details and results of the study.

The charter for the SCL study team included six specific tasks to be completed as part of the study. Following is a list of the tasks which were to be addressed and a summary of the results of each of these tasks.

1. *Review and evaluate existing SCL capabilities.*

The team studied SCL to learn its capabilities and then considered how it would be used in the JPL environment.

2. *Identify the applicability of SCL to the JPL mission set and type, e.g. planetary missions with significant one-way light times.*

It was determined that SCL could be used in the JPL environment. The conversions between various time bases would be handled by ground software.

3. *Identify required extensions to and shortcomings of SCL's existing capabilities and/or drivers for the development of a clone.*

SCL does not currently have native support for the concepts of privileged commands, critical sequences, or mark and rollback. There are mechanisms in SCL which could be used to provide capabilities which could be used to provide partial support in these areas, but these mechanisms might be awkward to use and would not provide a capability as complete as that currently used on JPL missions.

4. *Identify requirements and impacts on JPL flight software development as driven by an SCL implementation.*

An onboard SCL capability could be used for sequencing and most of the spacecraft fault protection routines. A real-time operating system would be required to support SCL. Uplink, telemetry processing, and attitude control tasks would not be done in SCL. A message-passing system would be required either as part of the real-time operating system or as a supplement to it.

5. *Identify requirements and impacts on ground software development as driven by an SCL implementation.*

Ground software support of SCL could be accomplished in three different modes. In all three modes, the mission planning tools up to but not including SEQ_GEN could continue to be used essentially as they are now. In the first mode, SEQ_GEN and SEQTRAN would be used as they are now except that SEQTRAN would use macros which produce SCL source code. In the second mode, SEQ_GEN and SEQTRAN would be replaced by a new program called SCL_GEN which would include many of the current capabilities of SEQ_GEN as well as a macro processor for support of blocks. In the third mode only the native SCL development tools would be used. This mode does not appear to be feasible for JPL missions currently envisioned.

6. *Document the results in a formal memorandum.*

The final report1 has been published.

Five items were identified which require further investigation. Following is a summary of these items.

1. *Determine how privileged commands, critical sequences, and mark and rollback will be handled.*

SCL does not have native support for privileged commands, critical sequences, or mark and rollback. These capabilities are considered essential for current JPL missions. A determination must be made as to which of these capabilities, if any, are essential for future JPL missions. For capabilities which are still required, suitable mechanisms must be defined within the context of SCL.

2. *Determine the ground software mechanism which will be used to generate onboard sequences.*

Two viable options have been identified for generating onboard sequences. One method uses SEQ_GEN as it currently exists and SEQTRAN with new macros to generate SCL source code. The other method uses a new program, SCL_GEN, which combines the functions of SEQ_GEN and SEQTRAN to generate SCL source code.

3. *Determine the method of command translation from mnemonics to bits.*

Command translation from mnemonics to bit could be accomplished by the Multimission Ground System Office (MGSO) ground software or by the SCL compiler. It appears to be advantageous to do the translation in MGSO software, but this choice needs to be reviewed.

4. *Determine the method of generating the SCL database.*

If the SCL compiler performs the mnemonic-to-bit translation, it may be necessary to include selected data from the Command Data Base in the SCL data base. If the translation is done by MGSO software, the interaction between the databases will be minimized.

5. *Determine the method of predicting events and monitoring spacecraft performance using event-driven sequencing.*

The use of event-driven sequencing will result in onboard events which will not be predictable in advance. A mechanism must be developed for monitoring such events without advance prediction.

The study concluded that two specific steps should be taken continue the investigation into identification of a language which can be adopted by JPL as a standard language for onboard commanding and sequencing.

1. *Develop prototypes of missions operations environments for two languages which are candidates for use by future JPL missions for onboard commanding and sequencing.*

2. *Identify additional candidate languages which could be used by future JPL missions for onboard commanding and sequencing.*

One of the major products of the SCL study was a list of requirements for candidate languages for use by JPL for a common onboard language for command and control. This list consists of fourteen requirements for a language which would support missions of the type which JPL has conducted to date. The list also contains fourteen additional requirements for a language which would support missions of the type which JPL is expected to conduct in the future. Such missions are expected to have more onboard autonomy and will require more onboard capability than has been provided on previous JPL missions. An appendix of the report contains the twenty-eight requirements and an evaluation of seven different languages which could be considered by JPL for future missions.

As a result of the SCL study and the two recommendations made by the study team, a followon effort has been funded at JPL. The objective of the followon task is to develop prototypes of two ground mission operations environments for two commercial process control languages which are candidates for use by future JPL missions for onboard commanding and sequencing.

One of the primary requirements for a language to be selected for the prototypes was availability as a supported commercial product. Because of the interest in SCL at JPL by Pluto Express and other projects, SCL was selected as one of the two languages for the prototypes at the commencement of the prototype task. None of the other six languages which were evaluated by the SCL study team met the requirements of being an adequately-supported commercial product. Thus the prototype design team initiated a search for a commercial language which could be used for the second prototype. Two languages were identified which were potential candidates: G2 by Gensym Corporation and RTWorks by Talarian Corporation. Both languages were demonstrated for the prototype design team.

Both G2 and RTWorks contain an inference engine and other capabilities required for consideration as a candidate for use by future JPL missions for onboard commanding and sequencing. However, the G2 implementation is monolithic and requires 32 MB of memory as well as extensive disk swap space for operation while the RTWorks is modular. Because the RTWorks inference engine provides capabilities required for onboard command and control and requires only a small fraction of the memory required by G2, RTWorks was selected as the language for the second prototype.

The prototype demonstration will consist of two separate systems, cooperatively developed, working together to demonstrate the capabilities required for a flight command and control system. The prototype task described above is responsible for developing a prototype ground mission operations environment which is capable of generating the data required for the onboard command and control system. The other task, funded separately, is the development of the two onboard command and control system prototypes to be programmed using SCL and the RTWorks inference engine.

The prototype ground system operations environments, shown in Figures 1 and 2, will utilize existing software developed by the JPL Multimission Ground Systems Office (MGSO). Specifically, the SEQ_GEN sequence generation program, the SEQTRAN sequence translation program, and the multimission command data base programs will be adapted to generate SCL and RTWorks code for the prototypes. For both prototypes, a Spacecraft Activities File (SASF) will specify the sequence of activities desired. The SASF will be input to SEQ_GEN which will produce a Spacecraft Sequence File (SSF). The SSF will be input to SEQTRAN which will produce source code for SCL or RTWorks. SEQTRAN is a macro-driven program. Two sets of SEQTRAN macros will be written to translate the input SSF to source code: one set to generate SCL code and one set to generate RTWorks code. The multimission command translator will provide information used to translate each specific command as required.

The ground software processing differs after generation of the SCL or RTWorks source file, depending on which language is used. For SCL, the source code is compiled by an SCL compiler into SCL object code. For RTWorks, the source code is used directly by the onboard interpreter. After compilation of SCL code or generation of RTWorks code, the resulting file is packaged for simulated uplink.

The prototype of the onboard command and control system to be used in the demonstration is being built in the JPL Flight System Testbed (FST). The Flight System Testbed at JPL provides an environment for testing new hardware and software for future JPL missions. Flight System Testbed personnel have developed a generic spacecraft simulation which can host specific components of new systems. A task in the Flight System Testbed is currently developing an onboard avionics system prototype for future JPL missions. The command and control component of the avionics system in the Flight System Testbed will be programmed in both SCL and RTWorks and used in conjunction with the prototype ground mission operations environment for demonstrating the use of SCL and RTWorks for spacecraft command and control.

Thus the combined prototype demonstration will use the prototype ground mission operations environment to generate programs for the onboard command and control system, the simulated uplink in the JPL Flight System Testbed to transfer the files to the simulated spacecraft using protocols of the Deep Space Network (DSN), and the prototype avionics system in the simulated spacecraft in the JPL Flight System Testbed to execute the onboard software generated by the prototype ground mission operations environment.

Both the SCL and RTWorks tasks are currently in progress. Both of the demonstrations are scheduled to be completed by September, 1996. After both demonstrations have been completed, a comparative evaluation of the two languages will be performed. Also, other candidate languages will be identified and considered for possible use in future prototypes.

In addition to the comparison of onboard languages, the ground system used to generate the onboard code will be reviewed for possible improvements.

Finally, a new effort is in progress to obtain the use of an actual spacecraft in orbit to demonstrate SCL and/or RTWorks for command and control in a flight environment. A successful flight demonstration would qualify a language for use in actual future missions such as New Millennium and Pluto Express.

REFERENCE

1. John A. Rohr, et. al., Spacecraft Command Language (SCL) Study Report, Jet Propulsion Laboratory, Document W4 MOSO9753-01-00, October, 1994.

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space
Administration.

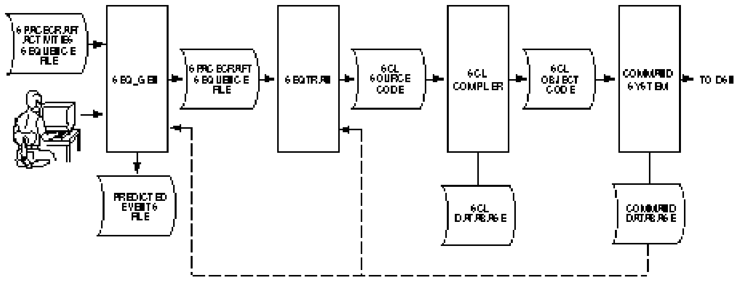

Figure 1: SEQUENCING AND COMMANDING WITH SCL

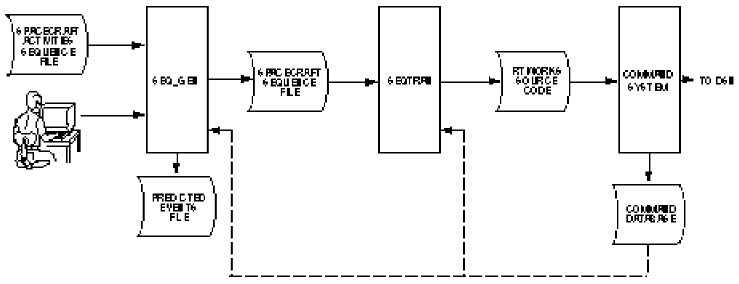

Figure 2: SEQUENCING AND COMMANDING WITH RTWORKS